(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
7 April 2016 (07.04.2016)

WIPO | PCT

(10) International Publication Number
WO 2016/053083 A1

(51) International Patent Classification:
G06F 17/30 (2006.01)

(21) International Application Number:
PCT/MY2015/050114

(22) International Filing Date:
1 October 2015 (01.10.2015)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
PI2014002834 2 October 2014 (02.10.2014) MY

(71) Applicant: **MIMOS BERHAD** [MY/MY]; Technology Park of Malaysia, Bukit Jalil, Kuala Lumpur, 57000 (MY).

(72) Inventors: **YONG, Keh Kok**; c/o Mimos Berhad, Technology Park of Malaysia, Bukit Jalil, Kuala Lumpur, 57000 (MY). **MAT NOR, Fazli Bin**; c/o Mimos Berhad, Technology Park of Malaysia, Bukit Jalil, Kuala Lumpur, 57000 (MY). **CHUA, Meng Wei**; c/o Mimos Berhad, Technology Park of Malaysia, Bukit Jalil, Kuala Lumpur, 57000 (MY). **KARUPPIAH, Ettikan Kandasamy A/l**; c/o Mimos Berhad, Technology Park of Malaysia, Bukit Jalil, Kuala Lumpur, 57000 (MY).

(74) Agents: **HEMINGWAY, Christopher Paul** et al.; Marks & Clerk (Malaysia) Sdn Bhd, Unit 6, Level 20, Tower A, Menara UOA Bangsar, 5 Jalan Bangsar Utama 1, Taman Bangsar (MY).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

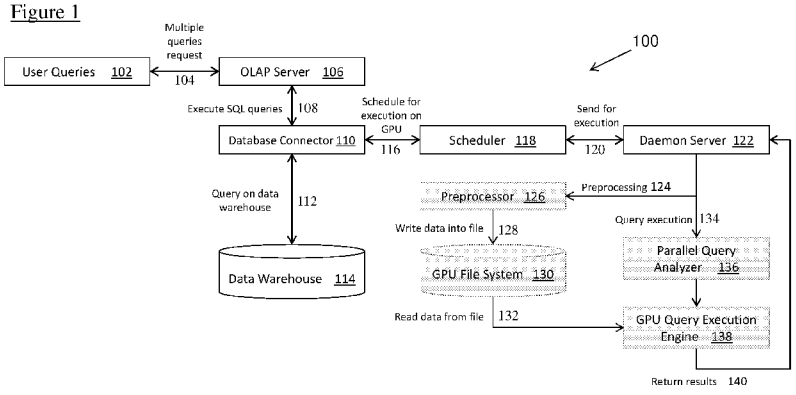
Published:

— *with international search report (Art. 21(3))*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*

(54) Title: SYSTEM FOR PROCESSING MULTIPLE QUERIES USING GPU

Figure 1



(57) Abstract: A system for processing multiple queries comprising a preprocessor (126) for processing input data into a preprocessed file structure (130) suitable for being processed by parallel instruction sets; a query analyser (136) for processing a plurality of input queries (602) into parallel instruction sets (340) which are grouped and rearranged into an optimized order; and a GPU query execution engine (138) for distributing the preprocessed input data across multiple GPU cores and executing the parallel instruction sets thereon.

WO 2016/053083 A1

SYSTEM FOR PROCESSING MULTIPLE QUERIES USING GPU

Field of Invention

The invention relates to a system for processing multiple queries in parallel using a Graphics Processing Unit (GPU).

Background

When dealing with large datasets, complex queries can take a long time to process. For example MultiDimensional Expression (MDX) is a specialised query language which can be used to query Online Analytical Processing (OLAP) databases, but processing is slow in conventional systems.

It is possible to convert an MDX query into multiple SQL queries so that these can be processed in parallel using a GPU, but existing systems do not do this efficiently, effectively using brute force to process the raw data as they are not optimized.

An aim of the invention is to provide an improved system for processing multiple queries in parallel using a GPU.

Summary of Invention

In an aspect of the invention, there is provided a system for preprocessing multiple queries comprising: a preprocessor for processing input data into a preprocessed file structure

suitable for being processed by parallel instruction sets; a query analyser for processing a plurality of input queries into parallel instruction sets; a GPU query execution engine for distributing the preprocessed input data across multiple GPU cores and executing the parallel instruction sets thereon; characterised in that the query analyser groups or rearranges the parallel instruction sets into an optimized order.

Thus as the data is preprocessed for the parallel instruction sets, and the parallel instruction sets are grouped into an optimized order, processing of the parallel instructions on the data is faster than compared to conventional systems.

In one embodiment the preprocessor extracts a table schema from the input data. Typically the preprocessor partitions the data.

In one embodiment the preprocessor obtains the configuration of the GPU cores. Typically the preprocessor creates a meta analysis profile which details the optimum data partitions for the configuration.

In one embodiment the input queries are SQL queries generated from MDX queries. Typically the query analyser decomposes the SQL queries into instruction sets and performs analysis to determine the shared input regions and result sets.

In one embodiment the query analyser reorders the instruction sets by analysing the dependency thereof and consolidating the independent instruction sets with shared input regions.

In one embodiment the query analyser tags the instruction sets with data usage by analysing the queries' utilisation of the output data region.

In one embodiment the query analyser tags the parallel instruction sets with GPU memory requirements.

In one embodiment the GPU query execution engine performs load balancing by distributing the preprocessed input data depending on the GPU cores configuration.

In one embodiment the GPU query execution engine checks each piece of GPU hardware in which multiple GPU cores are situated to determine if multiple instruction sets can be executed simultaneously thereon.

Brief Description of Drawings

It will be convenient to further describe the present invention with respect to the accompanying drawings that illustrate possible arrangements of the invention. Other arrangements of the invention are possible, and consequently the particularity of the accompanying drawings is not to be understood as superseding the generality of the preceding description of the invention.

Figure 1 is a schematic overview of an embodiment of the invention.

Figure 2 is a schematic diagram of the preprocessing stage

Figure 3 is a schematic diagram of the query analysis stage

Figure 4 is a schematic diagram of the query execution stage

Figure 5 illustrates an example of input data being converted to preprocessed data

Figure 6 illustrates an example of SQL queries being converted to parallel instruction sets

Detailed Description

With regard to Figure 1, there is illustrated a schematic overview 100 of an embodiment of the invention.

User queries 102 are submitted in the form of MDX requests 104 to an OLAP server 106. The OLAP server 106 may convert the MDX requests to SQL queries 108 for querying 112 the data warehouse 114 via a database connector 110. However, according to the invention, the SQL queries can be scheduled 116 for execution on a GPU using a scheduler 118 on a daemon server 122.

The daemon server includes a preprocessor 126 for preprocessing 124 input data into a preprocessed file structure suitable for being processed by parallel instruction sets. The preprocessed input data is written 128 onto disk in a GPU file system format 130 such that it can be read 132 therefrom when required.

In addition, the daemon server includes a query analyser 136 for processing 134 a plurality of input queries into parallel instruction sets, and a GPU query execution engine for distributing the preprocessed input data across multiple GPU cores and executing the parallel instruction sets thereon, returning 140 the results to the daemon server 122.

With respect to Figure 2, the steps performed by the pre-processor are illustrated in more detail. The preprocessor listens 202 for instructions from the daemon server and on receiving the same, determines 204 if any GPU resources are available.

Assuming GPU resources are available, data is extracted 206 from the data warehouse via the database connector, which provides data access between the database management system and the GPU parallel database system, and then a crumble analysis is conducted 208 to optimise parallelization of the input data.

In the crumble analysis 210 a table schema is extracted 212 from the data and the total of the column sizes of each table is computed 214. The configuration of the GPU devices is then obtained 216 and a meta analysis profile is created 218 which underlines the data partitioning and formulates optimum data independent partitions against the GPU capabilities. Large input data is thus partitioned into segments of efficient data communication size for the GPU parallelization based on the underlining hardware architecture.

A parallel execution plan is then created 220 comprising the steps 222 of: getting 224 the GPU device configuration, receiving 228 the meta analysis profile of the input data, and

determining 226 the optimum multi streams configuration by analysing the total streams usage and size of input data.

The data is then organised 230 into parallel streams in the GPU transport manager 232 which governs parallel transferring of data streams asynchronously between CPU and GPU via PCI express. Multiple streams are created 234 for segmented data compression. The data is split 236 into blocks and assigned to GPU threads. The parallel data is compressed 238 into streams and the data from multiple streams is synchronized 240. The subsets of output from the streams are then united 242.

The preprocessed data, now in an optimised GPU file system format, is then written 244 to disk, and once the plan has completed 246, the daemon server is notified 248.

With respect to Figure 3, the steps performed by the query analyser are illustrated in more detail.

Multiple SQL queries, converted from MDX queries, are received 302 from an OLAP server. A break down analysis is then performed 304, in which the multiple queries are decomposed 306 into instruction sets, the shared input region and result set is detected 308 and then written 310 into a shared input structure buffer as shared input data region indexes 312.

The resulting query instruction sets are then reordered 314 by analysing 316 the dependency thereof and assembling 318 the independent instruction sets with shared input region to provide reordered multiple instruction sets 320.

Parallel boundaries are analysed by examining the output data region 322, more specifically by analysing the boundaries of parallelization 324, generating memory boundary instruction sets 326 and computation boundary instruction sets 328, and analysing the process data usage 330 to provide multiple query instruction sets with data usage information 332.

Parallel instruction sets for GPU execution are then synthesized 334 by analysing 336 the order of the results sets (reordered multiple instruction sets 320 and multiple query instruction sets with data usage information 332) and composing 338 instructions from the combinations, to provide optimized multiple query instruction sets 340. These are then sent 342 to the parallel database engine.

With respect to Figure 4, the steps performed by the GPU query execution engine are illustrated in more detail.

The process starts 402 by receiving instructions sets 404 and generating 406 parallel APIs based on the instruction sets and outline tasks.

More specifically, the GPU memory usage is defined 408 by global, constant and shared memory. The GPU parallel thread utilization for blocks is then configured 410. If the GPU

supports dynamic parallelism 412, a pre-implemented dynamic query operator and dynamic kernel APIs are used 414, if not, a pre-implemented standard query operator and generic kernel APIs are used 418. A list of APIs for GPU parallel execution is generated 416 to form a task.

The tasks are inserted 420 into a worker queue, a fragment of input data from the GPU file system is read 422, and then executed 424 in the GPU by analysing 426 the streaming load balancing, performing 428 load balancing, then decompressing and computing streams 430 before merging and cohering 432 the streaming data.

The output of the results set from the GPU are united 434, and if there are no more tasks 436, the daemon server is notified 438 that the processing has completed and the results are returned thereto.

With regard to Figure 5 an example of input data 502 being converted to preprocessed data 510 is illustrated, wherein the different columns of strings 504, integers 506 and floating point numbers 508 are separated into different chunks 510.

With respect to Figure 6 an example of SQL queries being converted to parallel instruction sets is illustrated.

A group 602 of SQL queries (#1, #2, #3) are transformed 304 into instruction sets 604. The instruction sets are then optimized 314 as described previously to create reordered instructions sets 320 which are then tagged with memory usages for parallel processing.

It will be appreciated by persons skilled in the art that the present invention may also include further additional modifications made to the system which does not affect the overall functioning of the system.

5

CLAIMS

1. A system for processing multiple queries comprising:

a preprocessor (126) for preprocessing input data into a preprocessed file structure (130) suitable for being processed by parallel instruction sets;

a query analyser (136) for processing a plurality of input queries (602) into parallel instruction sets (340);

a GPU query execution engine (138) for distributing the preprocessed input data across multiple GPU cores and executing the parallel instruction sets thereon;

characterised in that the query analyser (136) groups or rearranges the parallel instruction sets into an optimized order.

2. A system according to claim 1 wherein the preprocessor (126) partitions the input and extracts a table schema therefrom.

3. A system according to claim 1 wherein the preprocessor (126) obtains the configuration of the GPU cores and creates a meta analysis profile which details the optimum data partitions for the configuration.

4. A system according to claim 3 wherein the GPU query execution engine (138) performs load balancing by distributing the preprocessed input data depending on the GPU cores configuration.

5. A system according to claim 1 wherein the input queries (602) are SQL queries generated from MDX queries.

6. A system according to claim 1 wherein the query analyser (136) decomposes the queries into instruction sets and performs analysis to determine the shared input regions and result sets.

7. A system according to claim 1 wherein the query analyser (136) reorders the instruction sets by analysing the dependency thereof and consolidating the independent instruction sets with shared input regions.

8. A system according to claim 1 wherein the query analyser (136) tags the instruction sets with data usage by analysing the queries' utilisation of the output data region.

9. A system according to claim 1 wherein the query analyser (136) tags the parallel instruction sets with GPU memory requirements.

10. A system according to claim 1 wherein the GPU query execution engine (138) checks each piece of GPU hardware in which multiple GPU cores are situated to determine if multiple instruction sets can be executed simultaneously thereon.

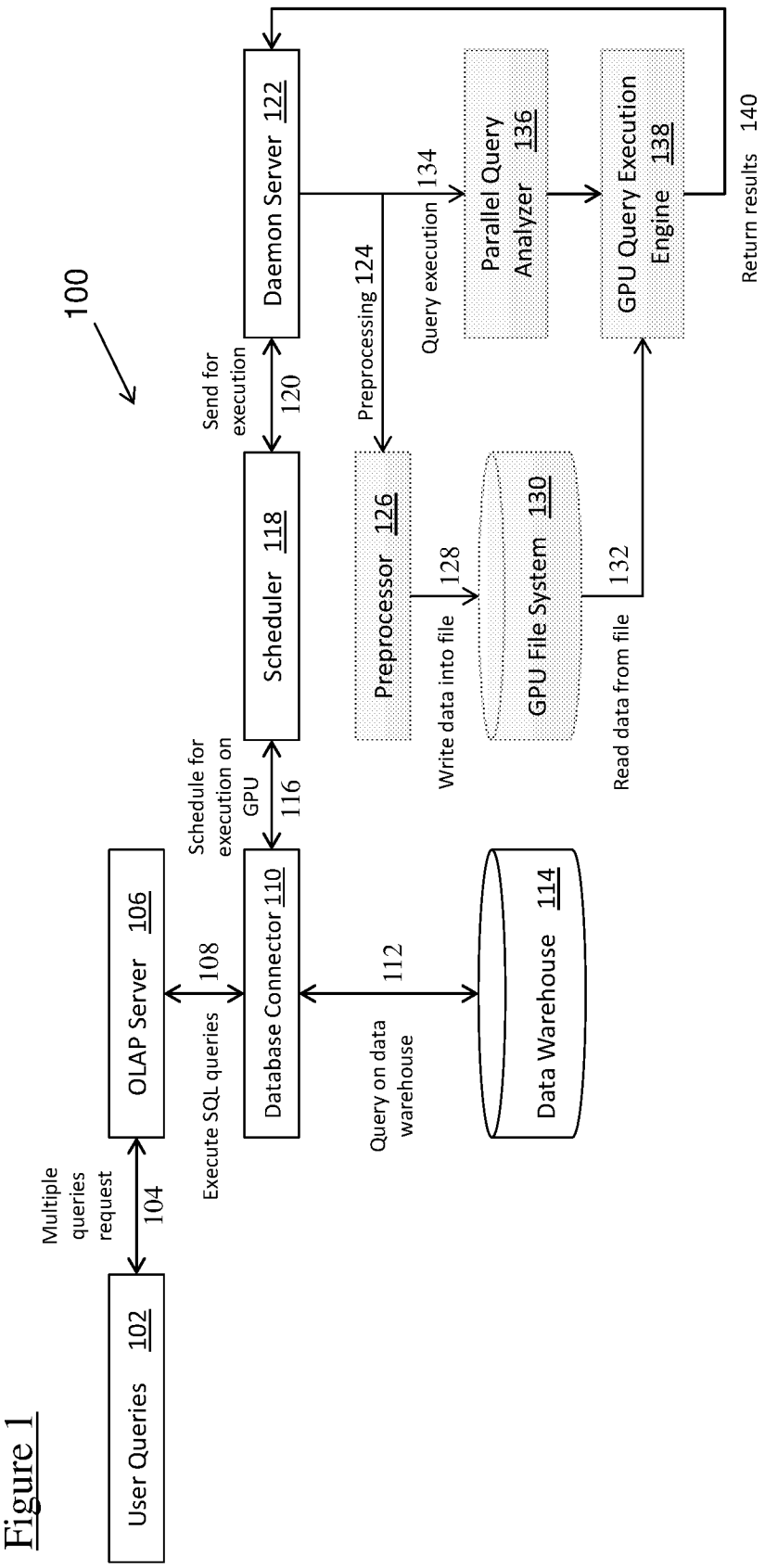

100
User Queries 102
Multiple queries request
104
OLAP Server 106
Execute SQL queries
108
Database Connector 110
Query on data warehouse
112
Data Warehouse 114
Schedule for execution on GPU
116
Scheduler 118
Send for execution
120
Daemon Server 122
Preprocessing 124
Preprocessor 126
Write data into file
128
GPU File System 130
Read data from file
132
Query execution
134
Parallel Query Analyzer 136
GPU Query Execution Engine 138
Return results 140

Figure 1

Figure 2

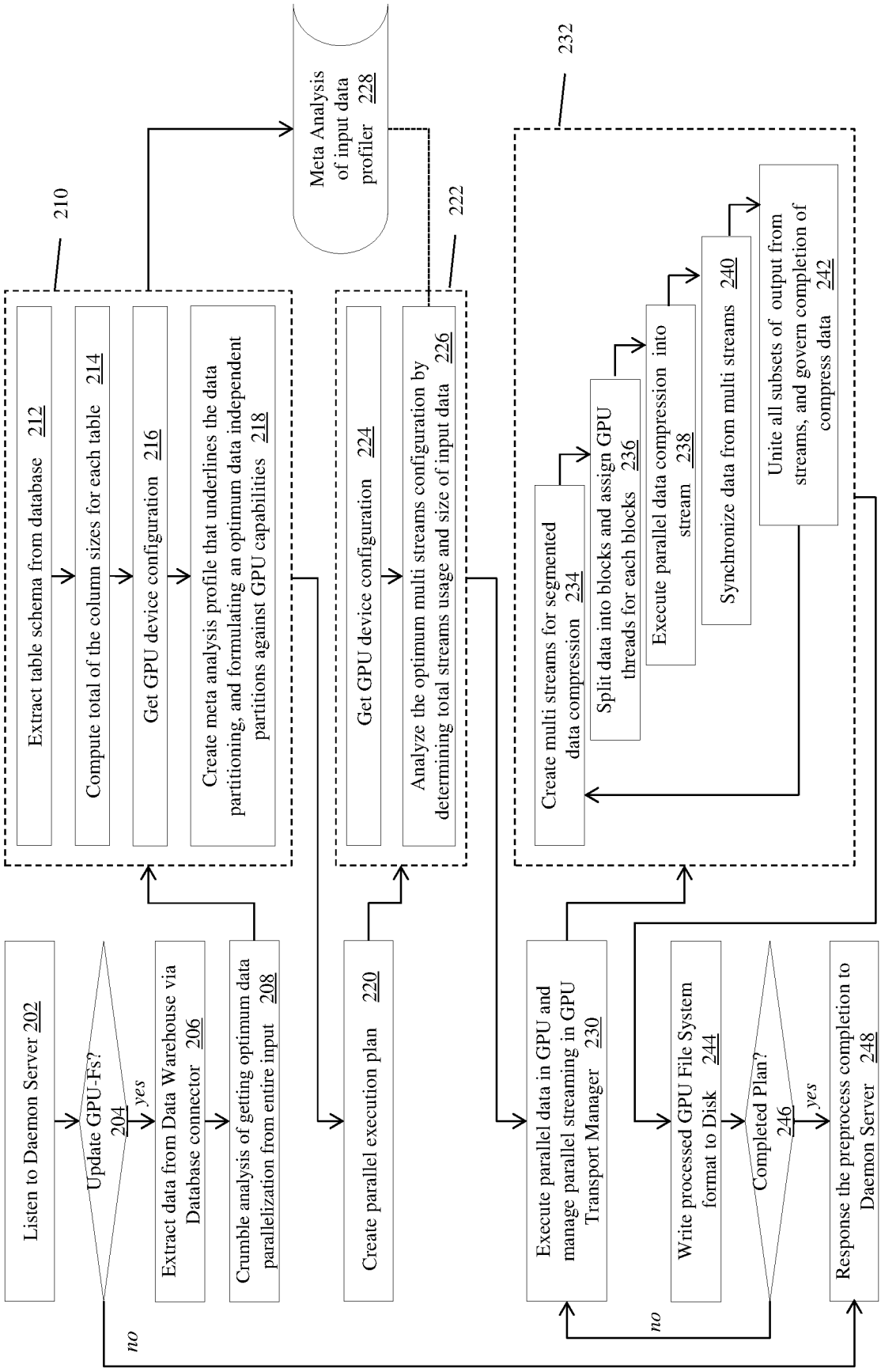

Listen to Daemon Server 202
Update GPU-Fs? 204
yes
no
Extract data from Data Warehouse via Database connector 206
Crumble analysis of getting optimum data parallelization from entire input 208
210
Extract table schema from database 212
Compute total of the column sizes for each table 214
Get GPU device configuration 216
Create meta analysis profile that underlines the data partitioning, and formulating an optimum data independent partitions against GPU capabilities 218
Meta Analysis of input data profiler 228
Create parallel execution plan 220
222
Get GPU device configuration 224
Analyze the optimum multi streams configuration by determining total streams usage and size of input data 226
232
Create multi streams for segmented data compression 234
Split data into blocks and assign GPU threads for each blocks 236
Execute parallel data compression into stream 238
Synchronize data from multi streams 240
Unite all subsets of output from streams, and govern completion of compress data 242
Execute parallel data in GPU and manage parallel streaming in GPU Transport Manager 230
Write processed GPU File System format to Disk 244
Completed Plan? 246
yes
no
Response the preprocess completion to Daemon Server 248

Figure 3

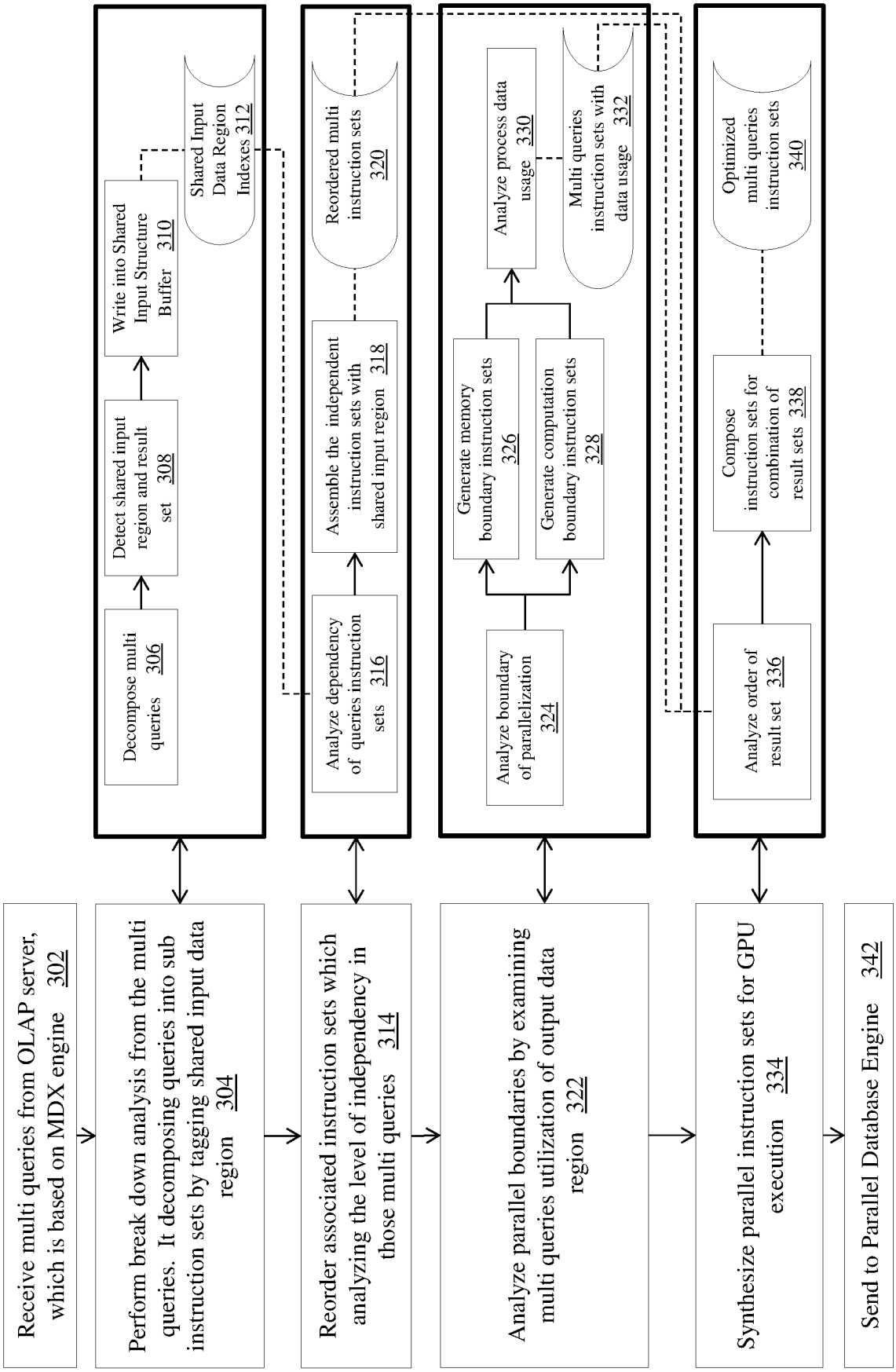

Receive multi queries from OLAP server, which is based on MDX engine 302
Perform break down analysis from the multi queries. It decomposing queries into sub instruction sets by tagging shared input data region 304
Decompose multi queries 306
Detect shared input region and result set 308
Write into Shared Input Structure Buffer 310
Shared Input Data Region Indexes 312
Reorder associated instruction sets which analyzing the level of independency in those multi queries 314
Analyze dependency of queries instruction sets 316
Assemble the independent instruction sets with shared input region 318
Reordered multi instruction sets 320
Analyze parallel boundaries by examining multi queries utilization of output data region 322
Analyze boundary of parallelization 324
Generate memory boundary instruction sets 326
Generate computation boundary instruction sets 328
Analyze process data usage 330
Multi queries instruction sets with data usage 332
Synthesize parallel instruction sets for GPU execution 334
Analyze order of result set 336
Compose instruction sets for combination of result sets 338
Optimized multi queries instruction sets 340
Send to Parallel Database Engine 342

Figure 4

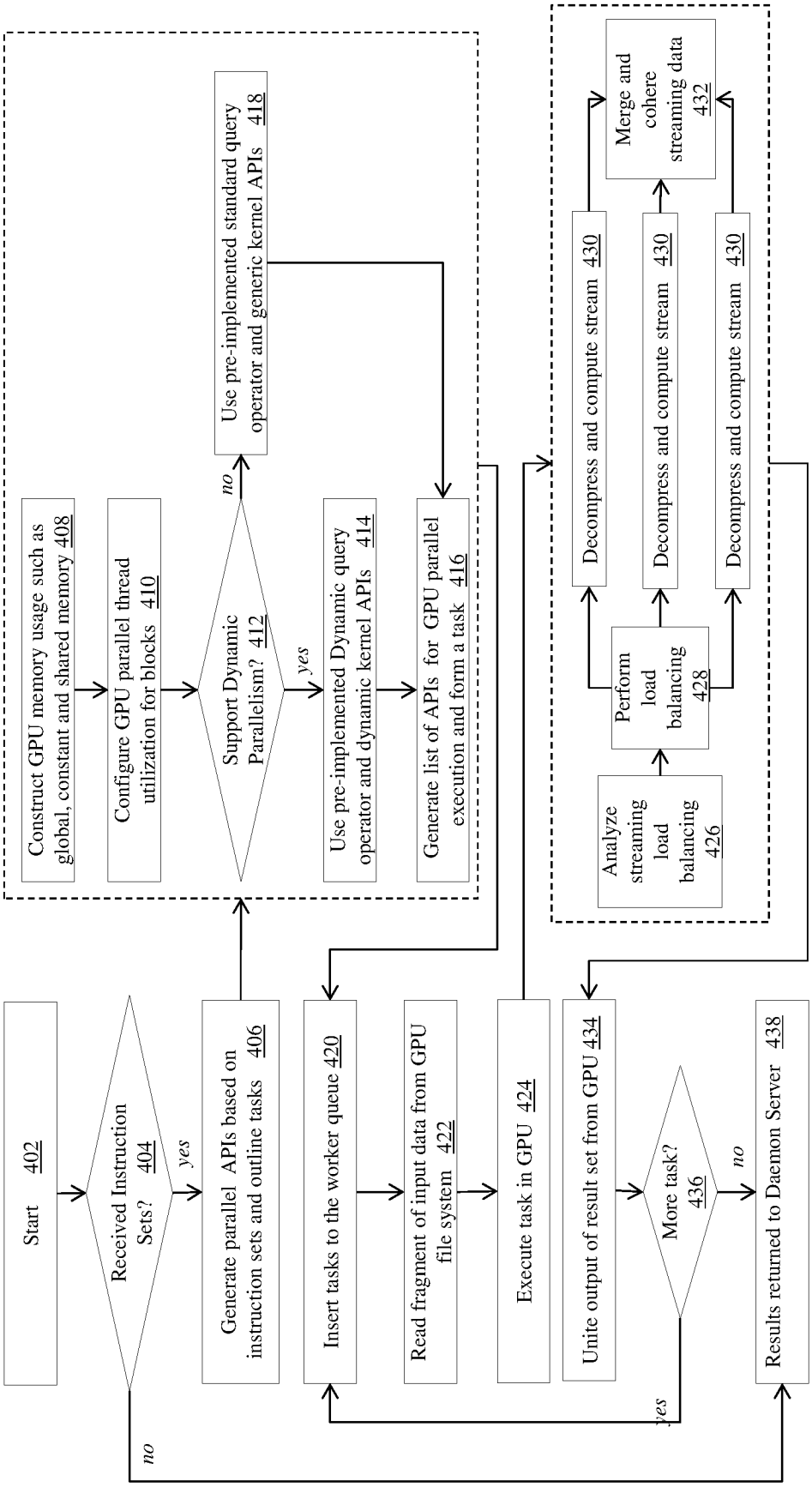

Start 402
Received Instruction Sets? 404
yes
no
Generate parallel APIs based on instruction sets and outline tasks 406
Construct GPU memory usage such as global, constant and shared memory 408
Configure GPU parallel thread utilization for blocks 410
Support Dynamic Parallelism? 412
no
yes
Use pre-implemented standard query operator and generic kernel APIs 418
Use pre-implemented Dynamic query operator and dynamic kernel APIs 414
Generate list of APIs for GPU parallel execution and form a task 416
Insert tasks to the worker queue 420
Read fragment of input data from GPU file system 422
Execute task in GPU 424
Analyze streaming load balancing 426
Perform load balancing 428
Decompress and compute stream 430
Decompress and compute stream 430
Decompress and compute stream 430
Merge and cohere streaming data 432
Unite output of result set from GPU 434
More task? 436
yes
no
Results returned to Daemon Server 438

Figure 5

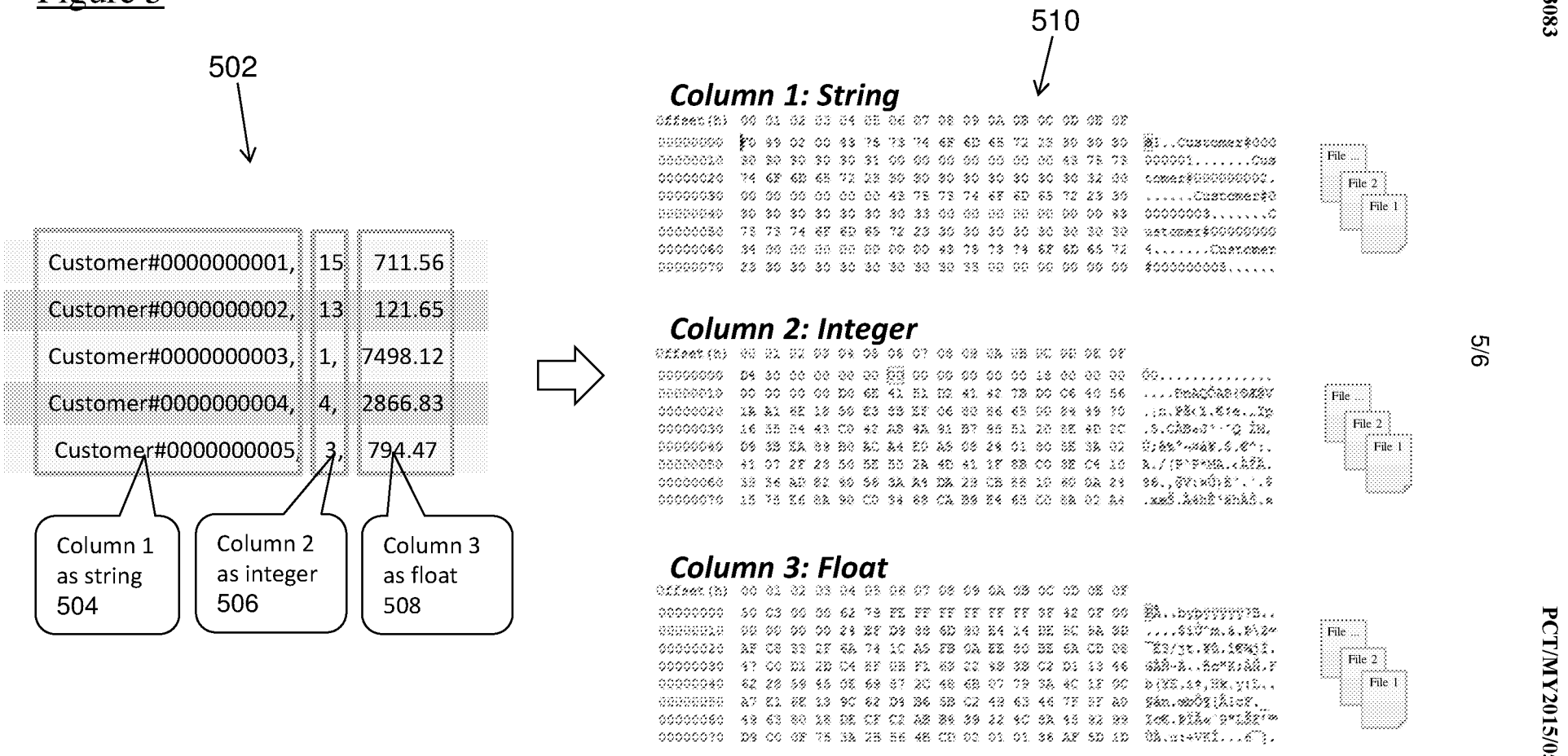
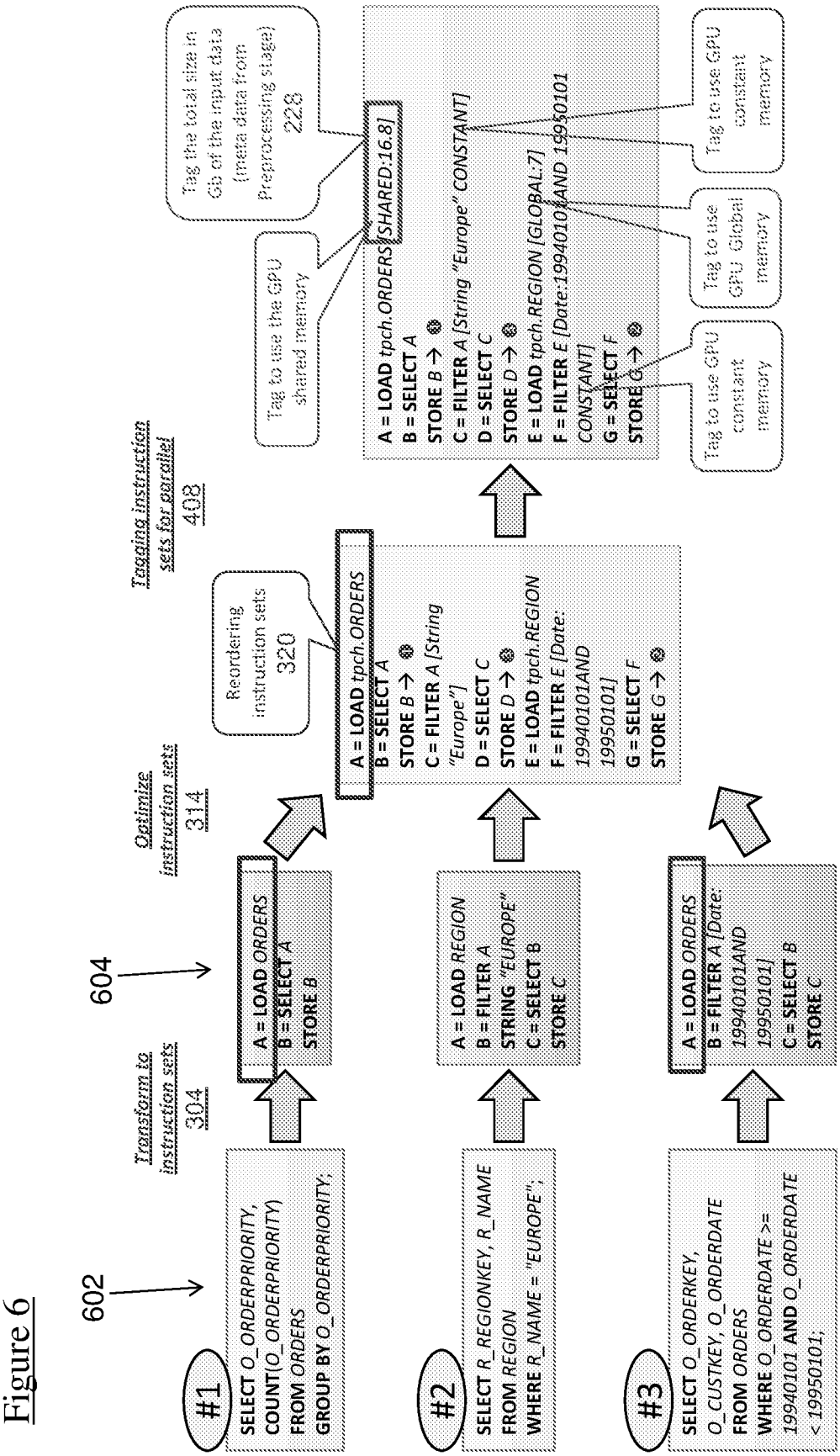

502
Customer#0000000001, 15, 711.56
Customer#0000000002, 13, 121.65
Customer#0000000003, 1, 7498.12
Customer#0000000004, 4, 2866.83
Customer#0000000005, 3, 794.47
Column 1 as string 504
Column 2 as integer 506
Column 3 as float 508
510
Column 1: String
Column 2: Integer
Column 3: Float
File 1
File 2
File ...

Figure 6


602
604
#1
#2
#3
SELECT O_ORDERPRIORITY, COUNT(O_ORDERPRIORITY) FROM ORDERS GROUP BY O_ORDERPRIORITY;
SELECT R_REGIONKEY, R_NAME FROM REGION WHERE R_NAME = 'EUROPE';
SELECT O_ORDERKEY, O_CUSTKEY, O_ORDERDATE FROM ORDERS WHERE O_ORDERDATE >= 19940101 AND O_ORDERDATE < 19950101;
Transform to instruction sets 304
A = LOAD ORDERS
B = SELECT A
STORE B
A = LOAD REGION
B = FILTER A
STRING "EUROPE"
C = SELECT B
STORE C
A = LOAD ORDERS
B = FILTER A [Date: 19940101AND 19950101]
C = SELECT B
STORE C
Optimize instruction sets 314
Reordering instruction sets 320
A = LOAD tpch.ORDERS
B = SELECT A
STORE B → ⊗
C = FILTER A [String "Europe"]
D = SELECT C
STORE D → ⊗
E = LOAD tpch.REGION
F = FILTER E [Date: 19940101AND 19950101]
G = SELECT F
STORE G → ⊗
Tagging instruction sets for parallel 408
A = LOAD tpch.ORDERS [SHARED:16.8]
B = SELECT A
STORE B → ⊗
C = FILTER A [String "Europe" CONSTANT]
D = SELECT C
STORE D → ⊗
E = LOAD tpch.REGION [GLOBAL:7]
F = FILTER E [Date:19940101AND 19950101 CONSTANT]
G = SELECT F
STORE G → ⊗
Tag the total size in Gb of the input data (meta data from Preprocessing stage) 228
Tag to use the GPU shared memory
Tag to use GPU constant memory
Tag to use GPU Global memory
Tag to use GPU constant memory

INTERNATIONAL SEARCH REPORT

International application No.
PCT/MY2015/050114

A. CLASSIFICATION OF SUBJECT MATTER

Int.Cl. G06F17/30(2006.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

Int.Cl. G06F17/30, G06F12/00

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Published examined utility model applications of Japan 1922-1996
Published unexamined utility model applications of Japan 1971-2016
Registered utility model specifications of Japan 1996-2016
Published registered utility model applications of Japan 1994-2016

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2008/0027920 A1 (MICROSOFT CORPORATION) 2008.01.31, [0027]-[0028],[0039]-[0042] & JP 2009-545060 A,[0015]-[0016],[0027]-[0030] & WO 2008/013632 A1 & EP 2044536 A1 & CA 2656024 A1 & KR 10-2009-0035545 A & CN 101496012 A & AU 2007277429 A1 & MX 2009000589 A & TW 200820023 A	1-10
Y	KAMIMURA,Junpei, Design and Evaluation of a GPU Accelerated Column Store Database, IPSJ SIG Technical Report 2011 August, 2011.08.15, pp.1-7, ISSN 1884-0930	1-10
A	JP 2008-165622 A (FURUYA,Toru) 2008.07.17, entire text; all drawings (No Family)	1-10

☐ Further documents are listed in the continuation of Box C.

☐ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
21.01.2016	02.02.2016

Name and mailing address of the ISA/JP	Authorized officer
Japan Patent Office 3-4-3, Kasumigaseki, Chiyoda-ku, Tokyo 100-8915, Japan	YOSHIDA,Makoto Telephone No. +81-3-3581-1101 Ext. 3599

5M 3659